SRC TR 88-51

Provably Good Parallel Algorithms For Channel Routing of Multi-terminal Nets

by

S. Krishnamurthy and J. Ja'Ja'

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **1988**	2. REPORT TYPE	3. DATES COVERED **00-00-1988 to 00-00-1988**

4. TITLE AND SUBTITLE **Provably Good Parallel Algorithms for Channel Routing of Multi-terminal Nets**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Maryland,Systems Research Center,College Park,MD,20742**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**26**	

Provably Good Parallel Algorithms For Channel Routing of Multi-terminal Nets

by

Sridhar Krishnamurthy *and* Joseph Já Já

Abstract

We consider the channel routing problem of a set of multi-terminal nets in the knock-knee model. We develop a new approach to route all the nets within $d+\alpha$ tracks, where d is the channel density, and $0 \leq \alpha \leq d$, such that the corresponding layout can be realized with three layers. Both the routing and the layer assignment algorithms have linear time sequential implementations. In addition both can be implemented on the CREW-PRAM model in $O(\frac{n}{p}+\log n)$ time, with p processors, $1 \leq p \leq n$, and n is the size of the input.

Provably Good Parallel Algorithms For Channel Routing of Multi-terminal Nets[1]

Sridhar Krishnamurthy

Department of Electrical Engineering

Systems Research Center

University of Maryland

College Park, MD. 20742

Joseph JáJá

Department of Electrical Engineering

Institute for Advanced Computer Studies

Systems Research Center

University of Maryland

College Park, MD. 20742

[1] Supported in part by NSA Contract No. MDA-904-85H-0015, NSF Grant No. DCR-86-00378 and by the Systems Research Center Contract No. OIR-85-00108

1. Introduction

Routing plays a central role in automated VLSI layout systems. This problem has been intensively studied in literature (e.g. [CJ],[MP],[P],[PL],[O],[RF]). Because of the combinatorial nature of routing, most of the corresponding optimization problems turn out to be NP-complete (for example see [S]). However good heuristics have been used effectively to generate good layouts. In this paper, we continue our research efforts in developing *efficient* parallel programming techniques to handle various routing problems. Our goal is to develop routing strategies that will result in parallel algorithms whose running time is $O(\frac{t(n)}{p} + f(p))$, where $t(n)$ is the best known sequential time, p is the number of processors, and $f(p)$ is a non decreasing function that reflects the routing cost on the given parallel model. The routing produced by these algorithms is expected to be as good as the best known sequential algorithms.

We consider the channel routing of multi-terminal nets in the knock-knee model. Provably good approximation algorithms (sequential) have been reported in [MPS],[SP] and more recently in [GK]. The basic strategy used is the well known greedy strategy applied either one column or one row at a time. However, it has been shown recently ([delaT]) that the routing produced by several variations of this strategy are P-complete, and hence there is little hope for parallelizing these strategies efficiently. We provide a new strategy which obtains provably good routing (which is in general different from those obtained in [SP],[MPS] and [GK] methods), such that the routing algorithms has a linear time sequential implementation. Moreover, the algorithm is fully parallelizable in the sense that it can be implemented on a Concurrent Read, Exclusive Write (CREW) PRAM model in $O(\frac{n}{p} + \log n)$ time with p processors, $1 \leq p \leq n$, where n is the size of the input. We are assuming that all terminals lie in the range $[1, N]$, where $N = O(n)$. A modified version of the algorithm will guarantee that the number of tracks is $d + a$, $0 \leq a \leq d$, where d is the density of the channel. In particular, for two terminal nets the modified version provides an optimal solution.

The rest of the paper is organized as follows. The basic definitions are described in the next section. The routing strategy is described in section 3, while in the last section we show the three layer wirability of the layout produced.

2. Definitions

We borrow some of the basic definitions of channel routing from [SP],[PL]. A net N is an ordered pair of integer sequences $((p_1, p_2, \cdots p_k), (q_1, q_2, \cdots q_h))$ where the p_i's are the *lower* terminals and the q_i's are the *upper* terminals. Without loss of generality, we assume that $k + h \geq 2$. If $k + h > 2$, then the net N is said to be a *multiterminal net*, otherwise N is a two-terminal net. An instance of a general channel routing problem (GCRP) is a channel consisting of rectangular grid, and a set of nets, each of which specifies a subset of terminals which lie on the grid points of the (horizontal) parallel boundaries. The goal is to route the wires such that the channel width is as small as possible.

Let $N_i = ((p_1^i, \cdots, p_{k_i}^i), (q_1^i, \cdots, q_{h_i}^i))$ be a set of nets. Let $l_i = min(p_1^i, q_1^i)$ and $r_i = max(p_{k_i}^i, q_{h_i}^i)$. The interval $[l_i, r_i]$ is a lower bound to the horizontal track demand of N_i. We can transform the GCRP into a fictitious two-terminal net channel routing problem, where each net N_i is replaced by N_i^* and l_i and r_i are referred to as the *left* and *right* terminals

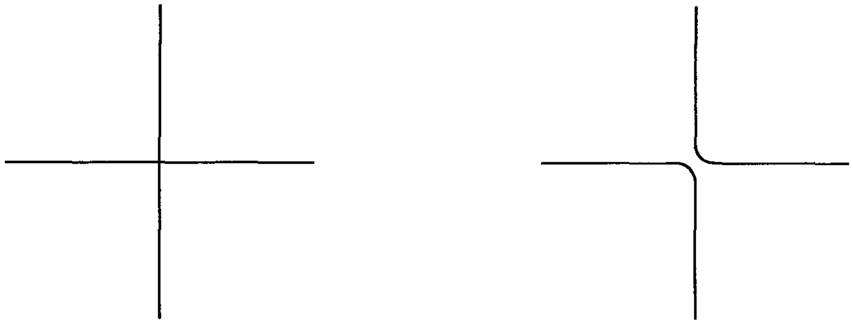


Figure 1: Types of shared grid point

of this fictitious net respectively. The *local density* d_x at x is defined to be the number of nets $[l_i, r_i]$ such that $l_i \leq x < r_i$. The density d is given by $d = max_x(d_x)$. It is clear that d is a lower bound for the minimum number of horizontal tracks and we call d the *essential density* of the GCRP.

A *wire-layout* (routing) in the knock-knee model consists of a set of edge-disjoint paths (made up of grid line segments) connecting the terminals of each net. Hence a shared grid point could be one of the two types (Figure 1).

Let $L_1, L_2, \cdots L_t$ be a set of conducting layers assumed to be stacked on top of each other in order, with L_1 on the bottom, L_2 next, $\cdots$ and L_t on the top. A contact between two layers called a *via* can be placed only at a grid point. A *wiring* of a wire-layout is an assignment of a single layer to each routing segment such that

1. No two segments of two distinct nets share a grid point on the same layer.

2. A routing path may change layers at a via.

3. No wire can use a grid point on a layer which is in between two layers with a via at

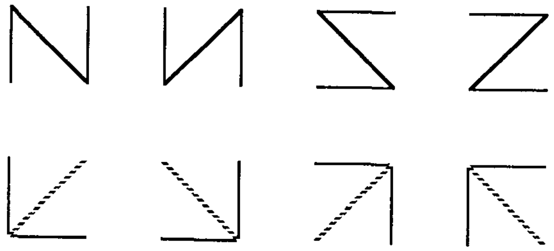


Figure 2: Forbidden patterns

that grid point.

It is known that any wiring in the knock-knee model can be done using four layers [BB] and that three layers suffice for the GCRP [SP]. Given a routing of a GCRP, the diagonal diagram can be obtained by representing a knock-knee at its vertex by a $\sqrt{2}$-length diagonal and a bend by a half-diagonal. On removing the half-diagonals we obtain the *core* layout. A wire-layout can be realized with three layers if its core can [PL]. A *partition grid* is defined as having vertices at points $(x+1/2, y+1/2)$ where x and y are integers on the channel grid. The edges of the partition grid are the segments connecting each vertex with its immediate horizontal, vertical and (45-degree) diagonal neighbors. A *legal partition* of a wire-layout is defined as a collection P of edges of the partition grid such that

1. Every internal vertex of P is incident with an even number of edges in P.

2. The diagonal edges in P are exactly the diagonals in the diagonal diagram.

3. P does not contain any of the *forbidden patterns* depicted in Figure 2.

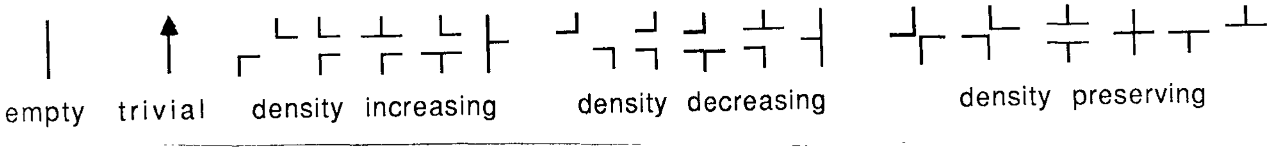


Figure 3: States of a column

We say a net N_i is *active* in the *vertical section* $(c, c+1)$ for a column c if $l_i \leq c \leq r_i$. Any vertical line x such that x lies between the interval $(c, c+1)$ cuts N_i in at least one point. Each such intersection of N_i is referred to as a *strand* of N_i. The type of terminals within a column (entry, exit or continuing terminals) define the *state* of the column. The state of a column can be of **type** *empty, trivial, density preserving, density decreasing or density increasing.* All the possible states of a column are shown below in Figure 3.

We use standard CREW (Concurrent Read Exclusive Write) shared memory model. All our results are stated in this model. However, our algorithms have fast implementations on fixed-interconnection networks such as the mesh or the hypercube. For example, all the algorithms stated in this paper can be implemented on a $\sqrt{n} \times \sqrt{n}$ mesh in time $O(\sqrt{n})$, where n is the input length.

3. Channel Routing

Given an instance of a GCRP of essential density d, our goal is to determine a wiring of all the nets within $2d$ tracks. In addition, the resulting layout should be realizable in three layers. Towards the end of this section, we shall briefly describe a modification of the wire-layout algorithm, leading to an improved layout, which uses at most $d+a$ tracks, where $0 \leq a \leq d$. Specifically, for two-terminal nets, $a = 0$, and the resulting layout is an optimal one.

The algorithm developed in [SP] produces the layout column-by-column. The overall strategy is similar to the approach of the 'greedy router' of [RF]. Unfortunately this approach seems to be inherently sequential. Our method is quite different and consists of the following main steps:

1. Create two sets of chains S_u and S_l. Each set consists of a partition of the nets into d chains satisfying certain properties to be outlined later. In particular, the nets in each chain define a set of nonoverlapping intervals. Initially a net has two symmetric segments above and below the track $y = 0$.

2. Assign a track number from the upper d tracks in the channel to each chain in S_u and a track number from the lower d tracks to each chain in S_l. Then wire all the nets for all the columns simultaneously.

The algorithm produces a layout which maintains the following property.

Property 1. Any net N_i which is active in column c has two strands $y = t_1(i) > 0$ and $y = t_2(i) < 0$.

We shall summarize the algorithm *Create Chains* developed in [CJ] which partitions a set of nets into d chains, where d is the density of the corresponding CRP. This algorithm will be used later on to obtain the initial sets of chains.

Algorithm Create Chains

Input: terminals l_i's and r_i's of all the nets $N_1, N_2, \cdots, N_n$.

Output: d chains of nets, where d is the density of the CRP.

1. Mark all the terminals. For each left terminal l_i of a net N_i, set $p(l_i) = r_j$ such that r_j is the nearest right terminal of some other net to the right of N_i. If two such terminals exist, then pick the one whose corresponding net is of the same type as N_i. However if no such r_j exists, then set $p(l_i) = \emptyset$. Similarly define $p(r_i)$ for each right terminal.

2. If $p(l_i) = r_j$ and $p(r_j) = l_i$ then set $succ(N_j) = N_i$ and unmark r_j and l_i. Create a reference point k between r_j and l_i.

3. Let $R_1, R_2, \cdots, R_m$ be the intervals determined by the reference points. For each R_i create lists $L(R_i)$ and $R(R_i)$ consisting of all the marked left and right terminals in R_i.

4. Create links between corresponding terminal pairs in $R(R_i)$ and $L(R_{i+1})$. Unmark all those terminals thus linked and merge intervals R_{2i-1} and R_{2i}. Repeat this step until only one interval is left.

Lemma 1 [CJ]. The number of chains created by the above algorithm is exactly d, where d is the channel density. This algorithm can be implemented on a CREW-PRAM in time $O(\frac{n}{p} + \log n)$ with p processors, $1 \leq p \leq n$. $\square$

The above chains are then modified in Algorithm 'Modify Chains' [CJ] so that they have the following property. Let c be any column. Then either

1. c is empty, or
2. c contains only one entry terminal, or
3. c contains two entry terminals of nets N_i and N_j. Let $N_i =< c, b_i >$and $N_j =< t_j, c >$. The following two cases can then arise:

   - If N_i has an exit terminal and N_j has an entry terminal in c, then they both belong to the same chains and one is a successor of the other.
   - Suppose both N_i and N_j exit at c. The other case is dealt with similarly. Let $N_i' = succ(N_i)$ and $N_j' = succ(N_j)$. Then they either have their entry terminals on the same column or the column of N_i' or N_j' which is closer to c has only one entry terminal.

We will now outline each of the main steps of our algorithm. The algorithm below creates initial chains of nets in sets S_u and S_l which will be modified later to satisfy certain desired properties.

Algorithm Initial Chaining

Input: terminals l_i's and r_i's of all the fictitious nets $N_1^*, N_2^*, \cdots, N_n^*$.

Output: two sets S_u and S_l with each set containing a partition of the fictitious nets into d chains, where d is the essential density.

1. Using algorithm Create-Chains outlined above, obtain d chains where d is the essential channel density.

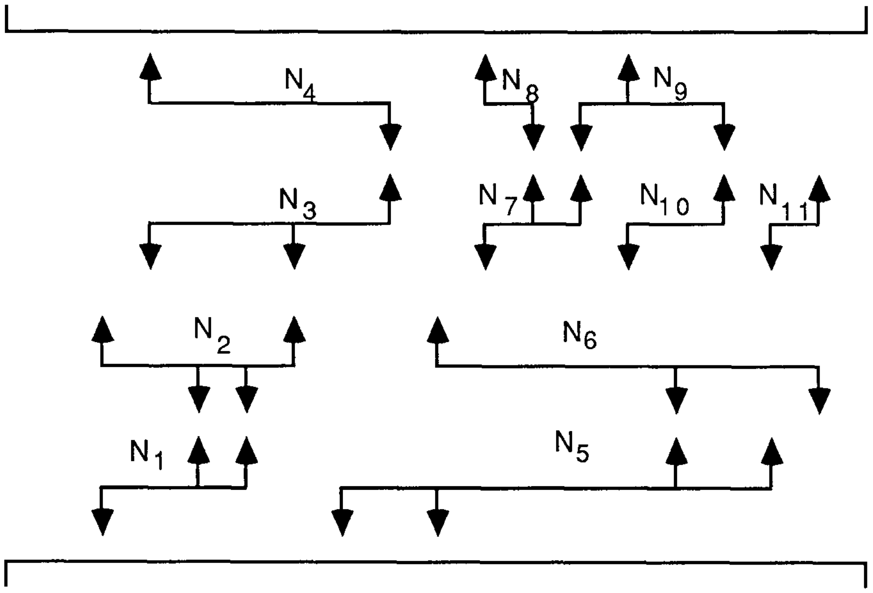


Figure 4: An instance of a GCRP

$N_1 \rightarrow N_8 \rightarrow N_{10}$

$N_2 \rightarrow N_5 \rightarrow N_{11}$

$N_3 \rightarrow N_7 \rightarrow N_9$

$N_4 \rightarrow N_6$

Figure 5: Chains created by Algorithm Initial Chaining

2. Duplicate the d chains thus obtained into two sets S_u and S_l each of which correspond to the upper and lower halfs of the channel. The chains in the two sets will be modified independently later on to satisfy certain properties.

As an example consider the general channel routing instance shown in Figure 4. The chains produced are shown in Figure 5.

Lemma 2. If there exists a column c containing an exit-terminal N_i and an entry-terminal N_j, then N_j is the successor of N_i in one of the chains. Property 1 will also be satisfied.

Proof. $succ(N_i) = N_j$ because of a direct result of algorithm 'Create-Chains'. In the first step of the algorithm, we have $p(l_j) = r_i$ and $p(r_i) = l_j$. Thus $succ(N_i) = N_j$.

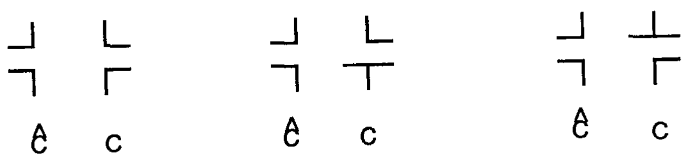
The chains in S_u correspond to the upper d tracks, while the chains in S_l correspond to the lower d tracks in the channel. Besides each net N is present in both the sets S_u and S_l. As a result each net would have been assigned two tracks, $y = t_1(N) > 0$ and $y = t_2(N) < 0$. Thus Property 1 is satisfied. □

The above chains can be used to wire all the nets in $2d$ tracks but the corresponding layout may not be realizable in three layers. So we have to modify the chains in both the sets S_u and S_l.

A column c, is said to be a *terminating* column, if

1. c has an exit terminal of N_i, or

2. c is the closest successor entry column of net N_i, and N_i has exit terminals in a column $\hat{c}$ whose *state* = $\begin{array}{c}\lrcorner\\ \urcorner\end{array}$ (see figure below).



All other columns are said to be *non-terminating.* We associate the pair $< c, N_i >$ to each terminating column, and refer to it as a *terminating pair.* A terminating pair $< c, N_i >$, is said to be an upper (lower) terminating pair if

- Net N_i has an exit terminal on the upper (lower) boundary in c or
- N_i terminates in c, and the successor of N_i is S_u (S_l) has its first upper (lower) terminal in c.

To satisfy the three layer wirability, the chains in the sets S_u and S_l are modified so as to satisfy the following property.

Property 2. Let c_i be any column. Then either

1. c_i is non-terminating, or
2. if c_i is associated with a lower (upper) terminating pair $< c_i, N_i >$, then either

   a. the column c'_i containing the first upper (lower) terminal of $N'_i = succ(N_i)$, in S_u (S_l) is non-terminating, or

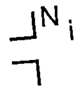
   b. column c'_i = column c_i.

The following algorithm outlines how to modify the chains S_u so that the above property holds. For modifying the set S_l we can essentially use the same algorithm with the obvious modifications.

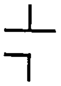
Algorithm Chain Modification

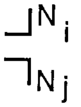
Input: state of columns and initial chain set S_u.

Output: new set of chains S_{unew} satisfying the property stated above.

1. If a column has an upper exit terminal of a net N_i, and is also the first upper terminal of the net, then delete N_i, from its chain in S_u, unless the column is of *state* = 

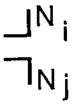
.

2. Mark all columns of $state(c) =$  .

3. Now consider all columns of $state(c) =$ 

. Two cases arise for such columns.

    a. Column contains an upper terminal labeled N_i, which is both the first as well as the last upper terminal. Let $\hat{N_i} = pred(N_i)$ in S_u. Modify S_u by setting the $succ(N_j) = N_i$ and $succ(\hat{N_i}) = oldsucc(N_j)$. Let c_i be the column to the right of c such that c_i contains the closest entry terminal of $succ(N_i)$ among its two successors. Mark c_i only if the entry terminal is a lower terminal.

    b. For the remaining columns of $state(c) =$ 

not considered in the previous step, we process as follows. Let c_i and c_j be the nearest columns to the right of c, such that they contain the entry terminals of the successors of N_i and N_j. Mark these columns if the entry terminals are the lower terminals.

4. For each marked column create an ordered pair $< c, c' >$ where c' contains the first upper terminal of $N_i' = succ(N_i)$ and N_i is the net which terminates in column c.

5. Group the pairs $< c, c' >$ into maximal groups $< c_0, c_1 >, < c_1, c_2 >, \cdots, < c_k, c_{k+1} >$. Let N_i denote the net which terminates in column c_i. Update the successors of these nets by setting the new successor of N_i to be the previous successor of N_{i-1} for all $0 < i \leq k$. In addition, set the new successor of N_0 to be the previous successor of N_k.

As an example consider the chains in Figure 5. The new set of chains created by the above algorithm are shown in Figure 6.

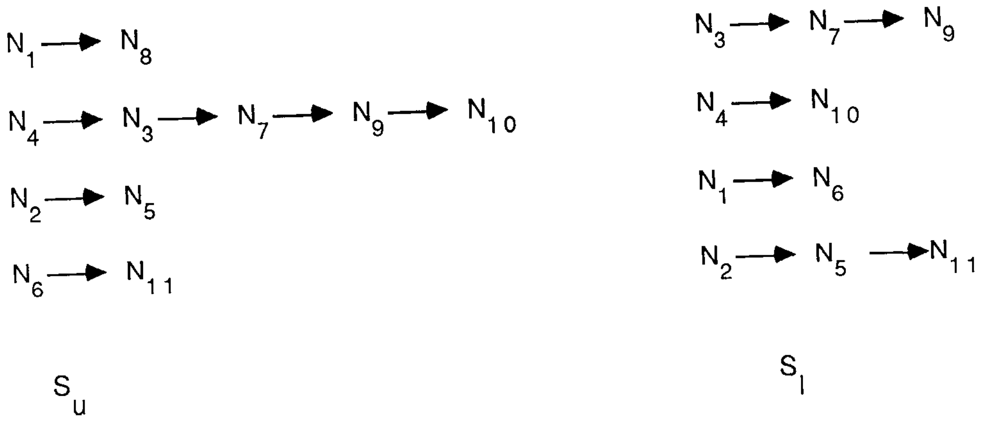


Figure 6: Modified sets of chains

Lemma 3. Algorithm 'Chain Modification' modifies the chains such that the new set of chains satisfies Property 2. Moreover the algorithm runs in $O(\frac{n}{p} + \log n)$ time with p processors, $1 \leq p \leq n$, on the CREW-PRAM model.

Proof. Without loss of generality we shall prove the lemma for the set of chains in S_u. If column c_i were non-terminating, then the property is satisfied trivially. Let us hence assume that the column c_i is terminating, associated with a terminating pair $< c_i, N_i >$. This results in the creation of an ordered pair $< c_i, c_{i+1} >$ in step 4 of the algorithm which is then grouped into a set of maximal groups in the subsequent step. Let $< c_i, c_{i+1} >$ be part of one such group, $< c_0, c_1 >, < c_1, c_2 >, \cdots, < c_k, c_{k+1} >$. For $i = 0$, the new successor of N_0 is the old successor of N_k, which has its first upper terminal at column c_{k+1}, which is non-terminating. For all other values of i, the new successor of N_i is the old successor of N_{i-1}, and thus the column which contains the first upper terminal of the successor of N_i is c_i itself and thus the property is seen to be satisfied. Notice that no new terminating columns are created.

The time and processor bounds of the algorithm can be easily established. □

After having obtained the modified sets of chains we proceed to do the wire layout as described in the algorithm Wire Layout. We denote $t_1(N_i)$ and $t_2(N_i)$ to be the tracks of N_i in S_u and S_l respectively. We can assign the track numbers $t_1(N_i)$ and $t_2(N_i)$, for all nets by a simple sorting step.

Algorithm Wire Layout

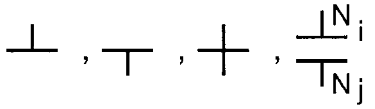
Input: state of the column, set of modified chains S_u and S_l.

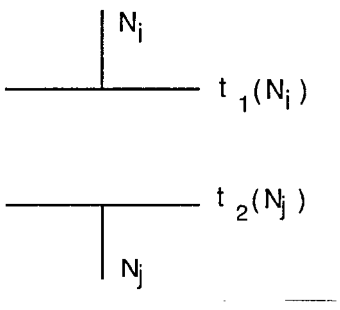
Output: description of the wire-layout to be performed in the column.

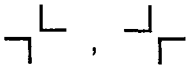
1. State of column is empty then do nothing.

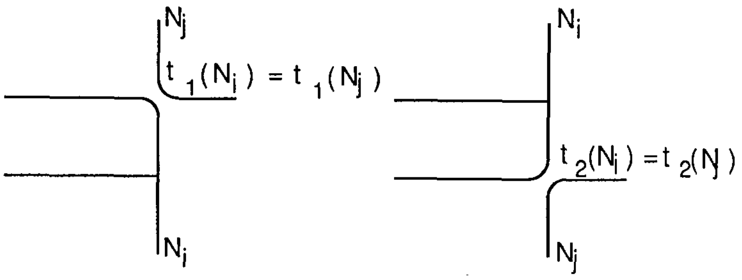
2. State of column is trivial then connect the upper and lower terminals.

3. State of the column is density preserving:

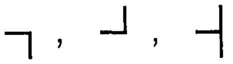
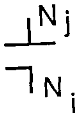
- *states*(c) =  are wired easily as shown below.

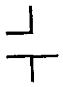


- *states*(c) =  where N_i is the terminating net and N_j is the starting net the wiring is as follows:

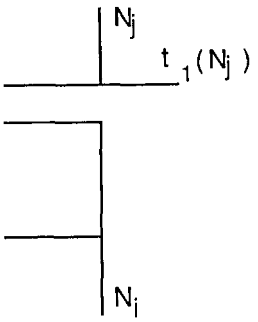


4. State of the column is density decreasing:

- If *state*(c) =  then wiring is simple and obvious.
- If *state*(c) = 

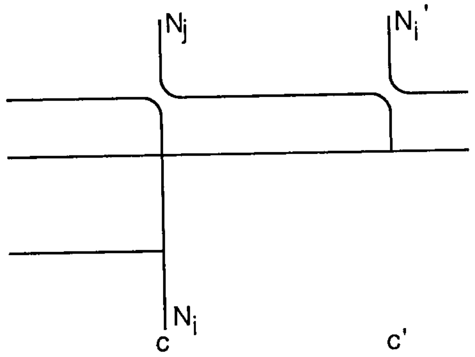
(*state*(c) = ) can be handled in a symmetric manner) then two cases may occur:

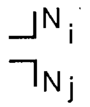
Case 1. $t_1(N_j) > t_1(N_i)$. Net N_i is terminated in the column as shown below.



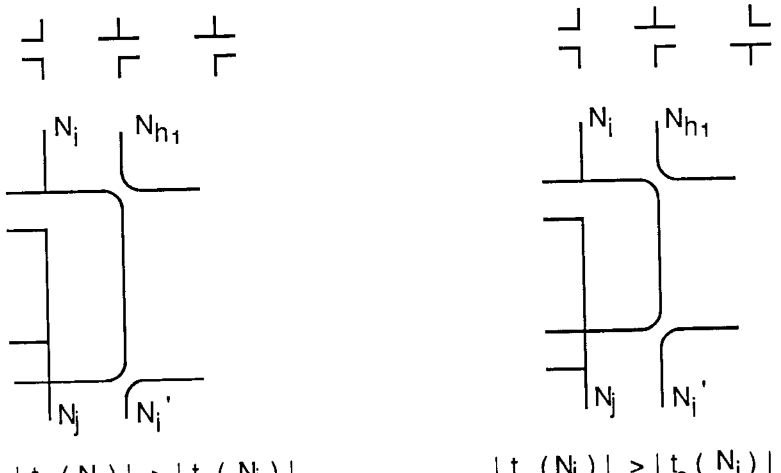
Case 2. $t_1(N_j) < t_1(N_i)$. While net N_i is terminated, net N_j is run temporarily in $t_1(N_i)$ until the nearest column e to the right of c such that column e has an upper terminal labeled:

- empty.
- N_j.
- N_i' where N_i' is the successor of N_i in S_u.



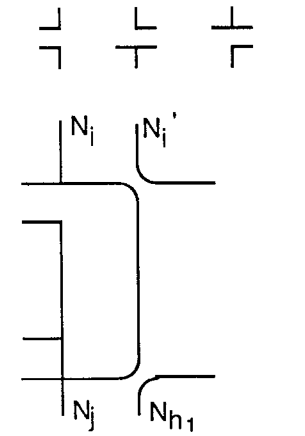
- If $state(c) =$  . Let $N_i' = succ(N_i)$ and $N_j' = succ(N_j)$. Let c_1 and c_2 denote the nearest columns to the right of c which contain the entry terminals of these nets respectively. Note that c_1 and c_2 are of type density increasing. Assume $t_1(N_i) > t_1(N_j)$. Other case can be handled similarly. The various orderings of c, c_1 and c_2 and their wirings are shown below:

O1.

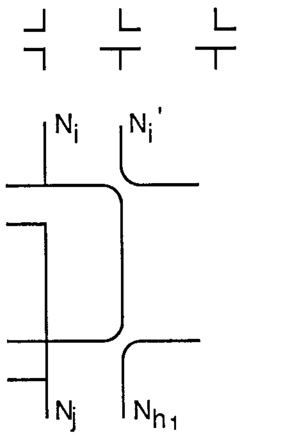


$|t_2(N_i)| > |t_2(N_j)|$ $|t_2(N_j)| > |t_2(N_i)|$

O2.

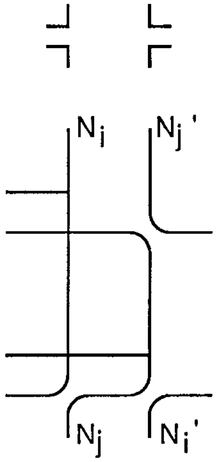


$| t_2(N_i) | > | t_2(N_j) |$

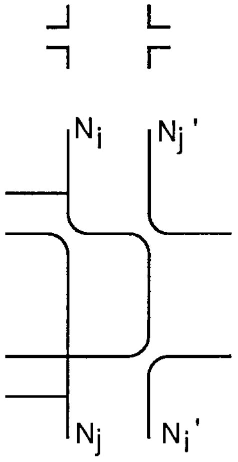


$| t_2(N_j) | > | t_2(N_i) |$

O3.



$| t_2(N_i) | > | t_2(N_j) |$



$| t_2(N_j) | > | t_2(N_i) |$

5. State of the column is density increasing: some of these columns have already been wired in the previous step. The remaining columns do not have any terminating nets and hence the wiring is done in the obvious manner.

Consider the example shown in Figure 4. The routing produced by the above algorithm is shown in Figure 7.

We notice that all the columns which are processed in step 2 of the algorithm will have a terminating net and hence these columns are the terminating columns.

Lemma 4. For a given instance of a GCRP algorithm Wire Layout will provide a legal

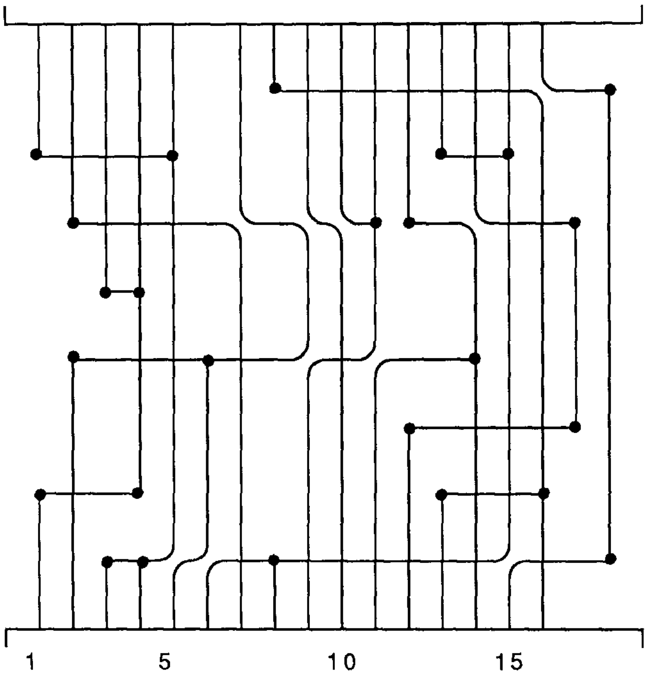


Figure 7: The layout generated by Algorithm Wire Layout

routing of all the nets using not more than $2d$ tracks for all the columns in the knock-knee model. □

Theorem 1. Given an instance of a GCRP of essential density d, it is possible to wire all the nets using $2d$ tracks in time $O(\frac{n}{p} + \log n)$ on a CREW-PRAM model with p processors, $1 \leq p \leq n$, where n is the size of the input.

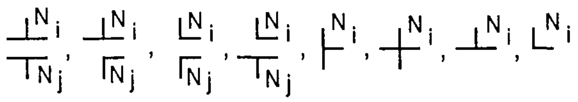
Proof: The theorem follows from the previous lemmas, and from the results of ([KRS]). □

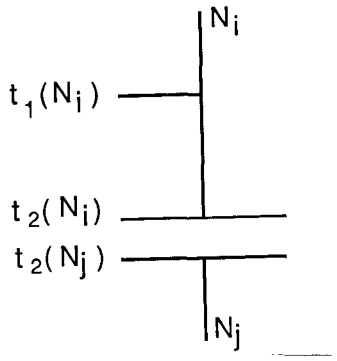
The main idea of the above algorithm was to establish an upper bound on the number of tracks used for any solution of a GCRP. As in the sequential version [SP], in conjunction with heuristics, we can develop efficient yet provably good solutions, for any GCRP. The solution uses $d + a$ tracks where $0 \leq a \leq d$. Additionally for a two-terminal net CRP, we can modify our algorithm to produce an optimal wire-layout using only the lower d tracks, where d is the density of the CRP.

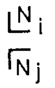
Modified Wire-Layout

The algorithm described above produces a routing which uses exactly $2d$ tracks, which can be realized in three layers, as shown in the next section. We now propose some modifications to the wire-layout strategy such that the resultant routing uses $d + a$ tracks, $0 \leq a \leq d$. Specifically for two-terminal nets $a = 0$, and the routing produced is an optimal one and is similar to the one produced in [CJ].

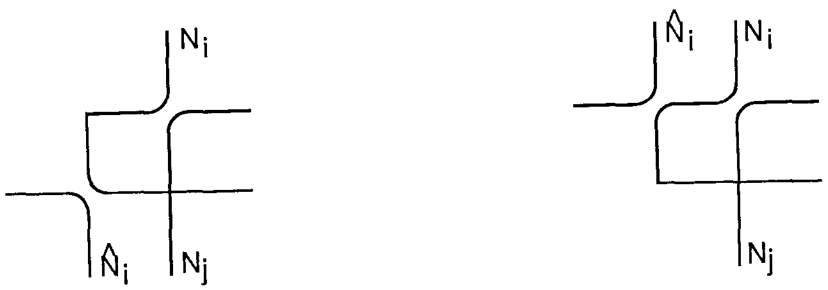
1. Any net which has terminals on only one boundary (say lower boundary), need only have a lower strand and an upper strand is unnecessary.

2. Consider a column c of one of the following states: *state*$(c) \in$ . If $|t_2(N_i)| < |t_2(N_j)|$, and the vertical ınterval between $t_1(N_i)$ and $t_2(N_i)$ is free in that column, then the net N_i is wired as shown below.



3. Consider a column c of *state*$(c) =$  such that $|t_2(N_i)| > |t_2(N_j)|$. Let c_i and c_j

be the columns which contain the exit terminals of $\hat{N}_i = pred(N_i)$ and $\hat{N}_j = pred(N_j)$. If the net having the exit terminals in the column nearest to the left of c, is a single strand net, the wiring for the various cases for column c below. The above modification leads us to the following result.



Theorem 2. Any channel routing routing problem with density d, can be routed in $d + a$ tracks for $0 \leq a \leq d$. Specifically for two-terminal nets $a = 0$. Besides the modified wire layout algorithm, runs in time $O(\frac{n}{p} + \log n)$ on a CREW-PRAM model with p processors, $1 \leq p \leq n$. □

4. Layer Assignment

The wire-layout produced in the first part of the previous section can be laid out in three layers. [PL] provides necessary and sufficient conditions for wiring a layout using three layers. The wiring can be done by finding a legal partition of the core of the diagonal diagram. The routing produced by our algorithm satisfies the following property:

Property 3. Any column irrespective of its type will have at most two knock-knees, with a diagonal \ on the bottom and a diagonal / above it.

By adding dummy diagonals if necessary, we can assume that each column is either empty or contains exactly two diagonals. Define a *reference line* as a vertical line that touches the end point of some diagonal. The diagonals touching a reference line, partition it into segments. We choose either odd or even vertical segments to obtain the legal partition.

We define the *constraint graph* as follows. The odd and even vertical segment choices of a reference lines are represented by v_{2i-1} and v_{2i}. Two vertices are connected by an edge if and only if the corresponding choices create a forbidden pattern. A forbidden pattern can be created only between adjacent reference lines.

Algorithm 'Select' in [CJ] selects a subset of vertices from the constraint graph, which will induce a legal partition of the wiring layout, assuming that the given graph does not contain any forbidden columns. The routing produced by our algorithm will have reference lines only of the types Figure 8. Each type is shown with its possible constraint graph. Notice that in none of the cases will there be any forbidden columns.

Theorem 3. Given an instance of a GCRP, it is possible to determine a three-layer assignment of the routing, in time $O(\frac{n}{p} + \log n)$ on a CREW-PRAM model using p processors, $1 \leq p \leq n$, where n is the size of the input. □

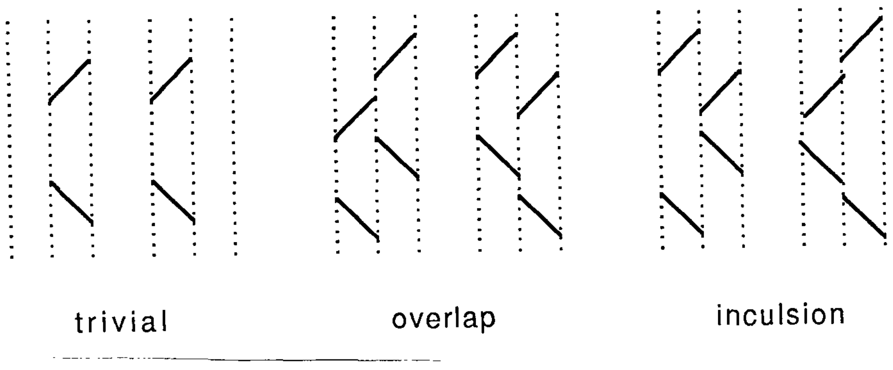


Figure 8: Types of reference lines

5. References

[CJ] Chang, S. and J. JáJá, " Parallel Algorithms for Channel Routing in the Knock-Knee Model," to be published in the International Conference on Parallel Processing, 1988.

[delaT] de la Torre, P., "On parallelism and some generalisations of the line packing problem," Unpublished manuscript 1988.

[GK] Gao, S. and M.Kaufmann, "Channel Routing of Multiterminal Nets," Proc. of the 19th Ann. ACM Symposium on Theory of Computing, pp.316-325, 1987.

[KRS] Kruskal, C., Rudolf, L., and M.Snir, " The Power of Parallel Prefix," IEEE Transactions on Computers, vol C-34 (10), pp.965-968, Oct. 1985.

[MP] Melhorn, K. and F. Preparata, "Routing Through a Rectangle," JACM, vol. 33(1), Jan. 1986, pp. 60-85.

[MPS] Melhorn, K., Preparata, F., and M. Sarrafzadeh, " Channel Routing in Knock-Knee Mode: Simplified Algorithms and Proofs," Algorithmica, 1986, pp.

213-221.

[O] Ohtsuki, T., "Layout Design and Verification," Advances in CAD for VLSI, vol. 4, North-Holland, 1986.

[P] R. Pinter, "River routing: methodology and analysis," Proceedings of the third CALTECH Conference on Very Large Scale Integration, March 1983, pp. 141-163.

[PL] Preparata, F. and W. Lipski, "Optimal Three-Layer Channel Routing," IEEE Trans. on Computers, C-33, 1984, pp. 427-437.

[RF] Rivest, R., and C. Fiduccia, " A Greedy Channel Router," Proceedings of the 19th Design Automation Conference, 1982, pp. 418-424.

[S] Sarrafzadeh, M., "Channel-Routing Problem in the Knock-Knee Model is NP-Complete," IEEE Transactions on CAD, vol. 6, 1987, pp. 503-506.

[SP] Sarrafzadeh, M., and F. Preparata, "Compact Channel Routing of Multiterminal Nets," Annals of Discrete Math., no. 25, April 1987, pp. 255-279.